Program #3
Due: Thursday Mar 20th, 2013 at 11:30PM

Instructor	Dr. Stephen Perkins
Office Location	ECSS 4.609
Office Phone	972-883-4522 (you can call me during office hours)
Email Address	stephen.perkins@utdallas.edu
Office Hours	Tuesday and Thursday after class (9:45AM to 10:30AM) and by appointment
TA	Indu Priya Kodali ixk130230@utdallas.edu Clark Center's CN1.202D Monday 10AM - 1PM Wednesday 5PM - 8PM

Purpose

Demonstrate the ability to create and use new classes, utilize class aggregation, and utilize Standard Template Library (STL) vectors.

Assignment

Rewrite and extend the code for Assignment 1 to utilize an STL *vector* of loan objects. You must create the loan class and define the appropriate methods to perform the required work. Your loan class must contain a vector that tracks payment history.

Your program should prompt the user for an input file name. The input file contains an unknown number of commands that will describe the operations to perform on the loans. Your program should open the file and perform all the commands listed in the file. Any output should go to the standard output. The input file will contain bad values, so you must perform checking and handle errors gracefully.

Some of the commands can change the value of the monthly payment or the interest rate of a loan object. You need to be able to print the entire history of payments from a loan, so you will have to track all payments to include their principal, interest, and payment amount. To print a full amortization schedule of a loan, you will print the entire history, and then calculate/print the remaining payments given the current state of the loan.

Deliverables

- You must submit your homework through ELearning.
- You must include your .cpp files, your header files, your input file, and an output file.
- Your code must exhibit the use of the STL vector class
- MUST use a multi-file project with your own header file.

Notes

No late homework is accepted.

Input File Format

The input file is a text file that contains an unknown number of **commands**. **Commands** are strings that determine what operations should be performed. If the commands have arguments, they are listed after the command. Multiple arguments will be comma separated. Spaces, tabs, and anything following // can be ignored.

The commands all have to do with keeping track of loans and printing information about loans. The commands are:

String Command	Description
cp	Calculate the payoff of a given loan. Takes one arg: int loanNumber;
n	Create a new loan to track. Takes three args: double initialLoanAmount double interestRate double monthlyPayment
mi	Modify the interest rate of a loan. Takes two args: int loanNumber double newInterestRate
mp	Modify the monthly payment of a loan. Takes two args: int loanNumber double newMonthlyPayment
pf	Print a full amortization of a loan (past plus future). Takes one arg: int loanNumber
rp	Receive payment. Should print split of principal and interest. Takes two args: int loanNumber double paymentAmount A payment may be more or less than the monthly payment. Excess payments should go straight to principal. Underpayments pay interest before principal.
sh	Show the History of payments for a loan. Takes one arg: int loanNumber

Your code must check for proper information for each command. Errors should be handled by printing a message and then ignoring the line. Examples of errors include, but are not limited to: unknown commands, commands without the proper number of arguments, commands with arguments that do not make sense, and commands that have the wrong types of arguments. It is recommended to try to read each line from the file as single string (command plus arguments plus comments plus whitespace) and then try to convert that line to a command with its appropriate arguments. If the conversion fails, then you can log the error. You can use the **strtok(),** **atof**(), and **atoi**() operations to help with this conversion.

An example file would be:

```
n 1000.00, 0.045, 125.00      // new loan  (starting loan is loan #1)
mi 1, 0.46                    // modify interest on loan 1
mp 1, 75.00                   // modify payment on loan 1
n 2000.00, 0.03, 175.00       // new loan  (loan #2)
n 10000.00, 0.023, 525.00     // new loan  (loan #3)
rp 1, 75.25                   // Receive payment for loan 1
pf 1                          // print full amortization for loan 1
sh 1                          // Show history loan 1
n 2000.00, 0.030, 78.00       // new loan (loan #4)
pf 4                          // print full amortization for loan 4
cp 4                          // calculate payoff loan 4
```